Executive Summary

Problem Statement

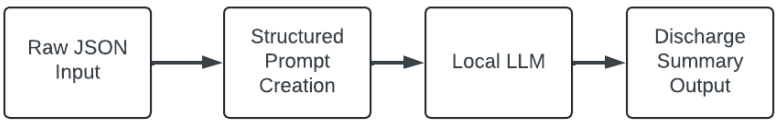
Physicians spend up to 33.4% of their time on non-patient-facing tasks, like writing discharge summaries, which limits their availability for direct patient care and contributing to inefficiencies, burnout, and communication gaps[1]. On average, doctors devote 8.7 hours per week to administrative responsibilities, many of which are critical yet repetitive[2]. One task is the discharge summary, which plays a vital role in ensuring continuity of care as patients transition from hospital to community settings. High-quality discharge documentation is essential for physicians to understand a patient's diagnosis, treatment course, medication changes, and functional status. Despite its clinical importance, discharge summary creation is time-consuming and often lacks consistency. This project addresses the need for efficient, standardized, and accurate discharge documentation by developing a secure, locally deployed prototype powered by Large Language Models (LLMs). The system automatically generates structured discharge summaries from inpatient data, ensuring privacy, reducing manual workload, and enhancing communication between providers and patients.

Technical Approach

The application is built on top of a directory of .py files and uses the Streamlit frontend to create a user interface that is easily understandable. The directory also contains a configuration json file which contains the boilerplate general instructions that will be customized on the basis of the patient data and any additional instructions provided by the user. For the LLM, the team chose to not use an external third party endpoint given the sensitive nature of patient details. Instead, the team used LM Studio to download a version of the LLama model to be hosted on a local device. The specific model used is Meta Llama 3.1 8B Instruct and the rationale behind this design choice is to ensure that the privacy of the patient is given utmost importance. Along with the different prompt engineering strategies used, the team experimented with the following pipelines to generate patient discharge summaries from raw inpatient data in JSON format.

1. **Direct JSON-to-LLM Pipeline**

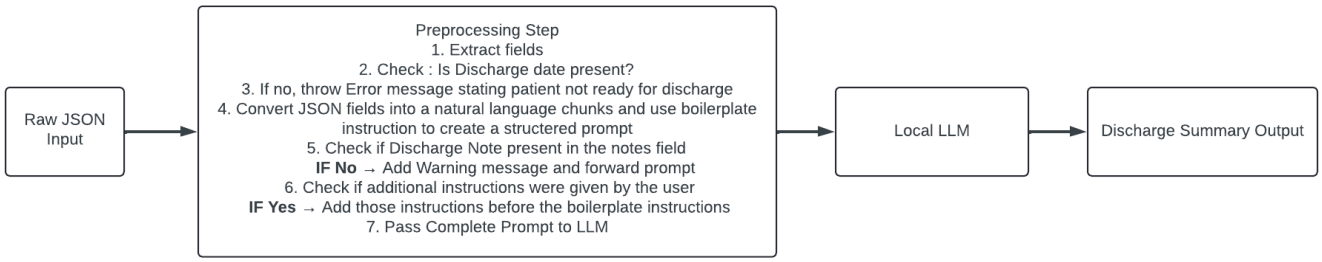
   The user directly entered the data JSON files and a prompt. The LLM would receive the raw JSON fields; this approach was the simplest of the three with minimal implementation, however, can result in low interpretability based on how well the LLM can parse through the JSON structure. Additionally, if the JSON file were to contain nested structures, the ability to parse this structure will become increasingly difficult, resulting in lower quality response.



2. **Conditional Preprocessing + LLM**

[1] Toscano F, O'Donnell E, Broderick JE, et al. How physicians spend their work time: an ecological momentary assessment. *J Gen Intern Med*. 2020;35(11):3166-3172.
[2] https://www.labmanager.com/administrative-work-consumes-u-s-physicians-time-and-erodes-morale-12802

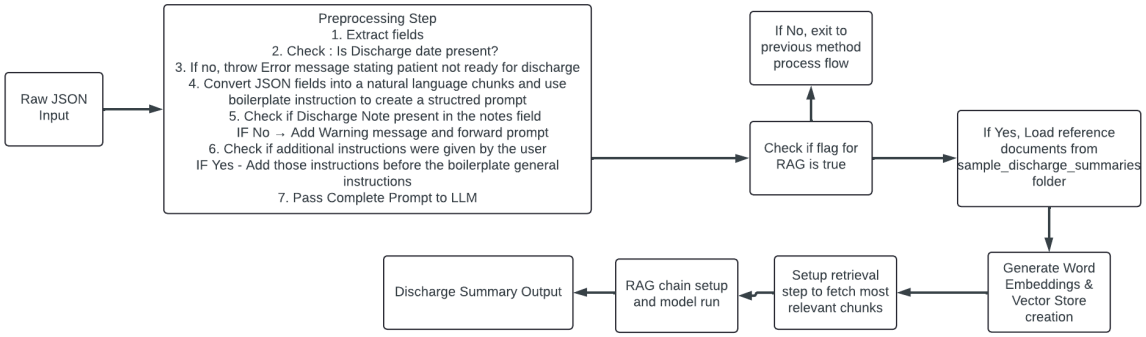
The user enters the data JSON file and the prompt. Instead of passing in the raw JSON data structure, the JSON file will first be pre-processed, formatted into natural language chunks. Additionally, at the pre-processing step, it checks if the discharge date is present. If so, this new pre-processed JSON data (along with the prompt) will be passed into the model. The more transparent inputs can lead to more coherent outputs and avoid unnecessary API calls. Additionally, if the patient data contains a discharge date but does not contain a discharge note in the Notes field of the JSON, the LLM is instructed to include a warning on top of the generated summary highlighting this fact. If the user provides additional prompts that are supposed to be included in the query, they are inserted on top of the general boilerplate instructions. The user also has the capability to edit the generated prompt before the call to the LLM will be made. Finally, the call to decide whether a patient is eligible for discharge is not completely left to the LLM but rather to the presence of a discharge note and discharge date. The presence of these values indicate that there has been a physician intervention which will overrule any decision that the LLM will take. There can be corner case scenarios which require discharge without the patient meeting the requirements of discharge according to the knowledge base of the LLM. However, the issue is that the pre-processing will only parse through the currently existing JSON data structure, therefore, it lacks flexibility.



3. **Retrieval-Augmented Generation (RAG) with LLM**

The third method involves the user providing a JSON file and a prompt as input. Additionally, they have the option to select a checkbox that enables referencing previously generated discharge summaries. Before processing, the JSON file is converted into natural language, as done in the previous method. The system then loads a set of sample discharge summaries and transforms them into vector embeddings. Each vector represents a chunk of the discharge summary text paired with its corresponding embedding. When the user enters their prompt, the prompt is also converted into an embedding. A similarity search is performed against the vector store to retrieve the top k most relevant chunks, with k = 4 as the default. These retrieved summaries are then combined with the user's prompt to create a richer and more context-aware input.

It is important to note that Retrieval-Augmented Generation (RAG) was not used to fine-tune the underlying language model itself. Instead, RAG-generated discharge summaries served as structural references to guide the formatting of outputs during generation. RAG was later utilized during the evaluation phase to help assess the quality and effectiveness of the prototype's responses.



Prompt Engineering Strategies

The prompt being used to generate the discharge summary for the patient employs the following prompt engineering strategies.

1. **Role Prompting**

   Role Prompting was used at the very beginning of the prompt when the LLM was assigned the specific role of a "junior clinical assistant on an internal medicine ward". This role definition is intended to anchor the LLM's response style and the clinical depth.

   "*You are a junior clinical assistant on an internal medicine ward, responsible for drafting accurate and comprehensive hospital discharge summaries. Your summary will be reviewed by attending physicians, shared with the patient's primary care provider, and used for clinical handover. It must clearly communicate the patient's course of treatment, clinical status at discharge, and follow-up plan. Base your summary solely on the patient data provided below, and adhere strictly to the formatting and clinical documentation guidelines that follow.*"

   The idea is to assign a clear professional identity and ensure that the model generates documentation which will be appropriate for clinical handover and not generate an informal summary. This practice also brings into consideration the target audience for the generated summary.

2. **Chain-of-Thought Prompting**

   The prompt instructed the LLM to narrate the patient's inpatient journey in chronological order, including daily trends in vital signs, laboratory results, clinical assessments, and treatments.

   "*Provide a thorough narrative of the inpatient stay. Include clinical assessments, investigations, interventions, treatment response, complications (if any), and the patient's clinical progress. Present this in chronological order, highlighting changes day by day.*"

   This sequential analysis of the patient data is enforced to reason through the patient's progress logically. By asking for explicit interpretation of clinical trends, the prompt encouraged multi-step reasoning in the narrative.

3. **Instruction Prompting**

   If the user provided additional instructions to be included in the prompt, they are supposed to overrule any general instructions that might be clashing. The following statement gets added to the prompt before the general instructions for the format of the discharge summary.

   "*The following instructions need to be followed. Overrule the general instructions if you have to. !!Give specific emphasis on clinical terminology*"

   For more context, the text after the !! is the additional specific instructions that the LLM is supposed to follow while generating the discharge summary.

4. **Output-Structured Prompting**

   The prompt provided extensive and explicit directions on how the LLM should format the content, including the requirement for full paragraphs, a minimum sentence count, privacy guidelines, clinical language standards, and a strict discharge safety check.

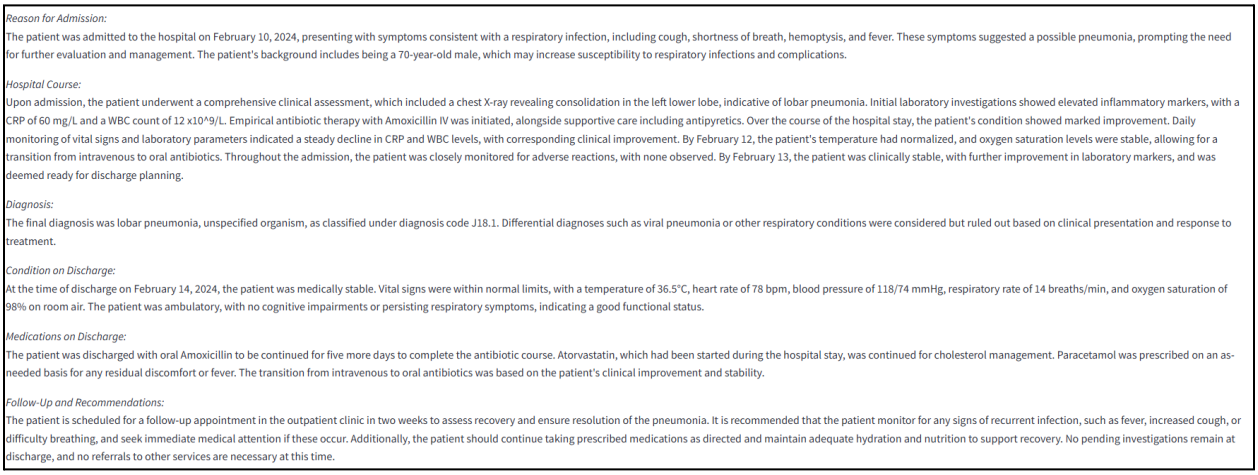
   "*Write each of the following sections as a clearly labeled header, followed by a detailed paragraph (no bullet points, no lists). The style should mirror that of formal clinical documentation.*"

   These instructions not only outlined what information to include, but also dictated how the LLM should organize, interpret, and present the information.

Insights

1. Direct JSON-to-LLM Pipeline

For the direct JSON-to-LLM pipeline, entering the just the JSON data directly will result in a generated discharge summary that follows this format: Reason for Admissions, Hospital Course, Diagnosis, Condition of Discharge, Medications on Discharge, Follow-up and Recommendation. This matches our base prompt structure.



Reason for Admission:
The patient was admitted to the hospital on February 10, 2024, presenting with symptoms consistent with a respiratory infection, including cough, shortness of breath, hemoptysis, and fever. These symptoms suggested a possible pneumonia, prompting the need for further evaluation and management. The patient's background includes being a 70-year-old male, which may increase susceptibility to respiratory infections and complications.

Hospital Course:
Upon admission, the patient underwent a comprehensive clinical assessment, which included a chest X-ray revealing consolidation in the left lower lobe, indicative of lobar pneumonia. Initial laboratory investigations showed elevated inflammatory markers, with a CRP of 60 mg/L and a WBC count of 12 x10^9/L. Empirical antibiotic therapy with Amoxicillin IV was initiated, alongside supportive care including antipyretics. Over the course of the hospital stay, the patient's condition showed marked improvement. Daily monitoring of vital signs and laboratory parameters indicated a steady decline in CRP and WBC levels, with corresponding clinical improvement. By February 12, the patient's temperature had normalized, and oxygen saturation levels were stable, allowing for a transition from intravenous to oral antibiotics. Throughout the admission, the patient was closely monitored for adverse reactions, with none observed. By February 13, the patient was clinically stable, with further improvement in laboratory markers, and was deemed ready for discharge planning.

Diagnosis:
The final diagnosis was lobar pneumonia, unspecified organism, as classified under diagnosis code J18.1. Differential diagnoses such as viral pneumonia or other respiratory conditions were considered but ruled out based on clinical presentation and response to treatment.

Condition on Discharge:
At the time of discharge on February 14, 2024, the patient was medically stable. Vital signs were within normal limits, with a temperature of 36.5°C, heart rate of 78 bpm, blood pressure of 118/74 mmHg, respiratory rate of 14 breaths/min, and oxygen saturation of 98% on room air. The patient was ambulatory, with no cognitive impairments or persisting respiratory symptoms, indicating a good functional status.

Medications on Discharge:
The patient was discharged with oral Amoxicillin to be continued for five more days to complete the antibiotic course. Atorvastatin, which had been started during the hospital stay, was continued for cholesterol management. Paracetamol was prescribed on an as-needed basis for any residual discomfort or fever. The transition from intravenous to oral antibiotics was based on the patient's clinical improvement and stability.

Follow-Up and Recommendations:
The patient is scheduled for a follow-up appointment in the outpatient clinic in two weeks to assess recovery and ensure resolution of the pneumonia. It is recommended that the patient monitor for any signs of recurrent infection, such as fever, increased cough, or difficulty breathing, and seek immediate medical attention if these occur. Additionally, the patient should continue taking prescribed medications as directed and maintain adequate hydration and nutrition to support recovery. No pending investigations remain at discharge, and no referrals to other services are necessary at this time.

However, there is an issue with entering the JSON data structure directly. In the example above, in the 'Medications on Discharge' section, neither dosage or frequency was provided; only information about the

names of the medications. This could be due to the inability to decipher the JSON data structure and lack of understanding the keys. The majority of information looks correct, however, is not as comprehensive as expected. Additionally, if the JSON data structure were to be more complex, the response will likely become more generalized.

2. Conditional Pre-processing + LLM & Retrieval-Augmented Generation (RAG) with LLM

Both methodologies represent a significant improvement over the previous approach, where the entire JSON was passed directly to the LLM. Sending the prompt along with the patient data to a local LLM server without additional context is operationally simple and lightweight. This method is most effective when the patient data is comprehensive and detailed. However, because it relies solely on the input prompt, it is more vulnerable to hallucinations and omissions of critical clinical information when the data is incomplete. Additionally, the generated summaries may vary in style and structure across different patients.

In contrast, the RAG-based approach improves the quality and consistency of outputs by first retrieving similar discharge summaries using FAISS and biomedical embeddings. Grounding the LLM's response in real-world examples leads to more accurate, complete, and stylistically consistent summaries. However, this method introduces additional complexity, including the management of embeddings, retrieval logic, and quality control of stored examples. It also results in slightly slower response times and creates more potential points of failure. Furthermore, the current RAG implementation retrieves from the full set of previously generated summaries, regardless of patient similarity. A more effective strategy would involve retrieving only those summaries associated with patients similar to the current case.

Limitations

It is very important to recognize the benefits of leveraging generative AI in order to create patient summaries, since it allows physicians to focus on more important tasks such as addressing patient problems during a medical visit. It is also important to recognize that with these benefits, there are some limitations that need to be taken into account for incorporating new generative AI technologies into health. One significant challenge that had to be taken into account is that in the healthcare industry it is crucial to comply with regulations like HIPAA which is essential for protecting patient's information from malicious purposes. This also helps the healthcare field maintain patient trust and uphold integrity of the healthcare system. By these means, it is important to first de-identify data before feeding it into our model, in addition, we run our model locally using LM studio since external API's could introduce risks such as data leakage. This allows the patient to keep information protected in a controlled environment. Another challenge encountered was that hosting the model locally is limited by the computational power of the current machine being used. This can lead to longer inferences, such as a response time of about 40 seconds unlike API endpoints which produce results in real time since they are hosted in servers that possess higher computational power. A delay in response can be a challenge for the users since it can affect their workflow and delay patient processes which is the opposite of the intended goal of improving efficiency among healthcare providers. Additionally, the model performance can change over time and could potentially stop following the latest medical guidelines. It is necessary to update it regularly in order to prevent the model from leaving out important details. This could affect a doctor's efficiency by spending more time in reviewing the generated document. In a similar light, the model can hallucinate and generate irrelevant and misleading information, which could potentially lead to confusion and mistakes. In order to reduce these limitations, it is important to take into account that the model needs to be continuously monitored and make sure it is secure to use.